“WinFS” And ADO.NET: Future Directions For Data Access Scenarios

Samuel Druker
DAT312
Development Manager
Microsoft Corporation

PDC05 DEVELOPER POWERED

Microsoft

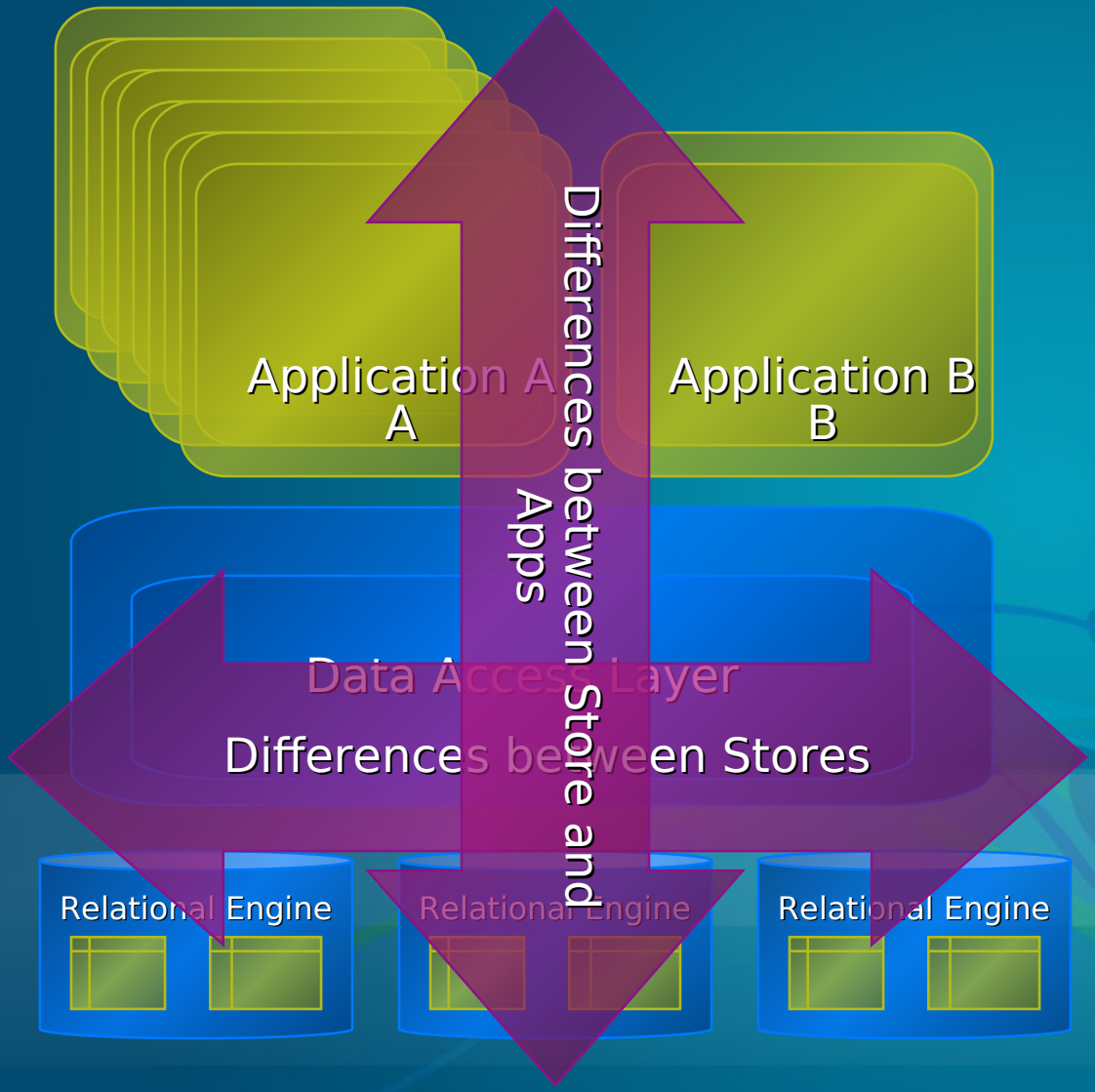
Agenda/Roadmap

- Innovation and Continuity
  - Data Access
  - Integrated Storage
- Evolution of ADO.NET
  - Today
  - Client-side Commands and Mapping
  - Object Mapping and Typed Connections
  - Language Integrated Query
  - WinFS
- Demo – a glimpse at the future

Innovation And Continuity

- Feedback from WinFS as of PDC03
  - Strongly typed object layer
  - Matching object-flavored query language
  - Completely symmetric file and row update patterns
  - New connection model for pathnames
- Migrate best of this model into the evolving state of the art for ADO.NET
  - More general purpose data model
  - Query language headed back toward SQL
  - LINQ integration
  - Componentized layering of functionality

Data Access Needs



- Differences between store and application growing
- Bridge differences between stores

DEMO

WinFS Programming With LINQ

PDC05
DEVELOPER POWERED

Samuel Druker
Development Manager
SQL Server

Why Improve Data Access?

- Codify existing good practice
  - Every application has a conceptual data model (somewhere)
  - Every application ships data out of tables into their (object-based) programming model
- Strong overlap in demands from deployed applications
  - Mapping and reshaping technology
  - Better query support
  - Smooth evolution from existing functionality
- Provide building blocks for productive development
  - Data model provides a lingua franca for component services that live close to the data
  - Drastic improvement in programming language integration

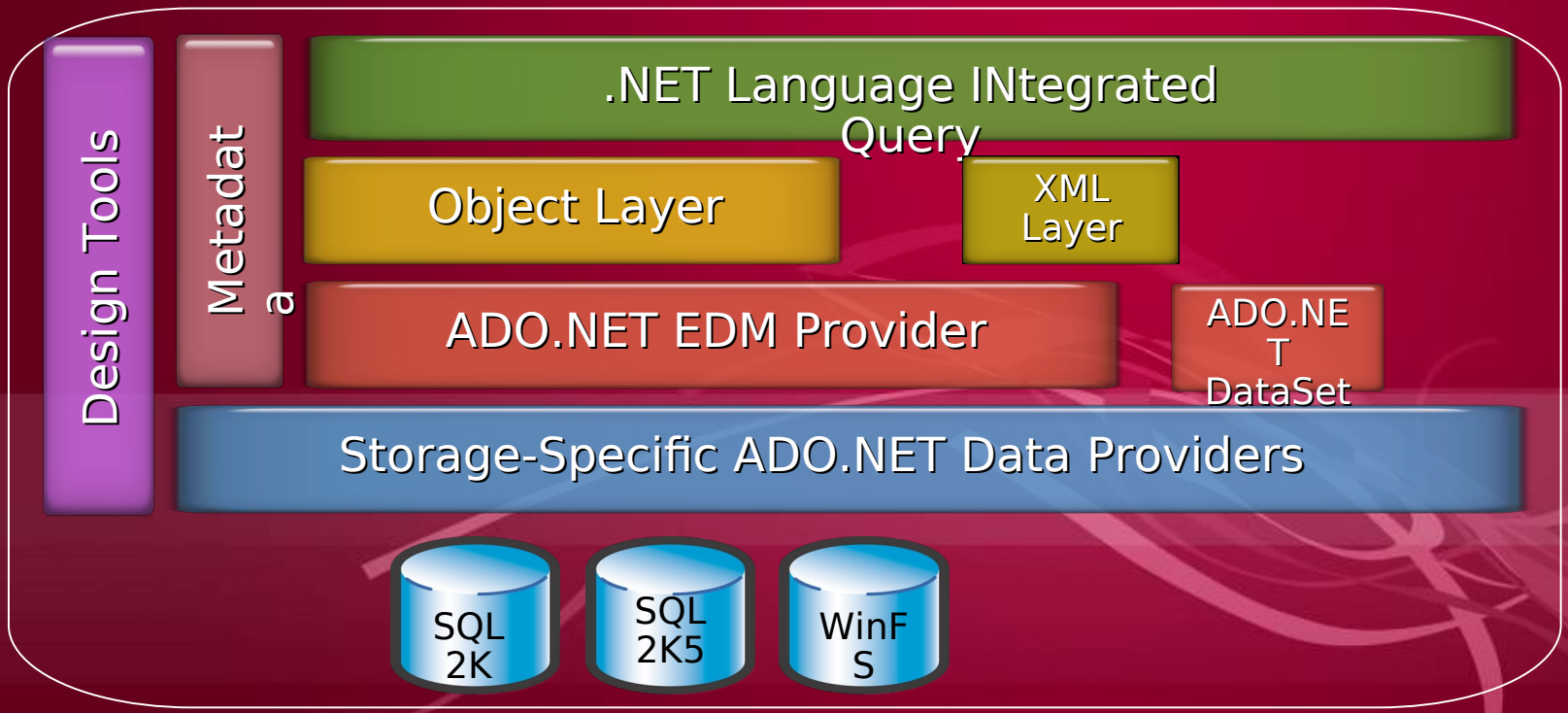
Layering The New Components

- Programming models with incremental richness
  - CLR objects
  - Structured entities
  - Reshaped data
  - Tables of rows
- Use what you need
  - “Pay-as-you-go” complexity in implementation
  - Consistent model for each additional layer

Integrated Storage

FUTURE TECHNOLOGIES

Data Access Architecture



What We Hear...

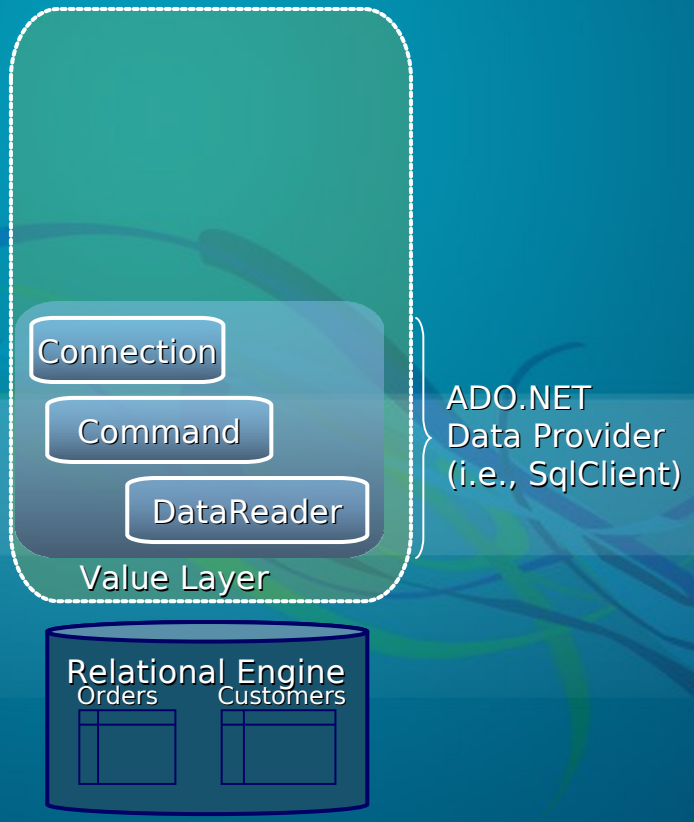
- “I want a better programming model for database apps...”
  - Evolution not revolution
  - Preserve investment in ADO.NET
  - SQL is the foundation
  - Bridge the gaps

ADO.NET Basics

- Basics for database development: Connections, commands and reader
  - Builds up from existing functionally in ODBC, OLE DB, ADO, and other shifts in programming models
- Provide a consistent component model
  - Provider, Connections, Commands
  - Supports disconnected model through DataSets

ADO.NET Today

```
using System.Data.SqlClient;

// Create and open a SqlConnection
SqlConnection sc = new
  SqlConnection(connStr);
sc.Open();

// Query for customers named "Smith"
SqlCommand customers = sc.CreateCommand();
customers.CommandText =
"SELECT ID, FirstName FROM Customers WHERE
  LastName='Smith'";
SqlDataReader rdr =
  customers.ExecuteReader();
while (rdr.Read())
  Console.WriteLine(rdr.GetString(1));
rdr.Close();

// Add a new customer
SqlCommand updatecmd = sc.CreateCommand();
updatecmd.CommandText =
"INSERT INTO Customers (ID, LastName)
  VALUES ('00', 'Ellis')");
updatecmd.ExecuteNonQuery();

// Close Connection
sc.Close();
```



What We Hear…

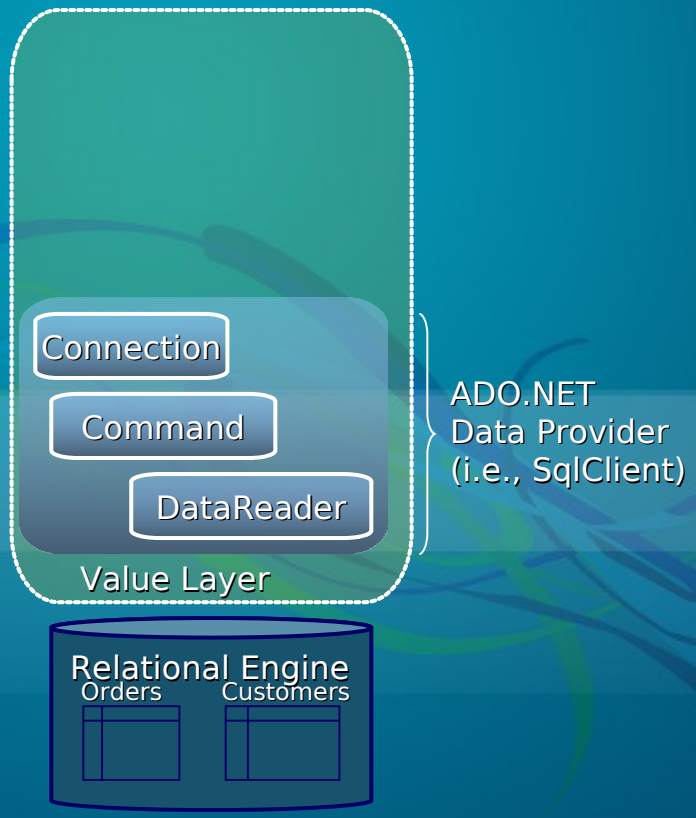
- “I want a better model for queries in my applications today…”
  - Provide real integrated query support up through my application
  - Help me avoid product specific query syntax
    - Insulate me from legacy systems
  - Don’t make me change the schema
    - I can’t…

Canonical Command Trees

- Structured, abstract representation of query
  - Encapsulates expression operators, parameters and updates
- Can be produced in various ways
  - Parsing of a textual query language
    - OPATH (WinFS beta, ObjectSpaces preview)
    - OSQL (new direction)
  - Generated from LINQ patterns
  - Public API surface for command trees
    - Query builder applications

Command Trees In Use

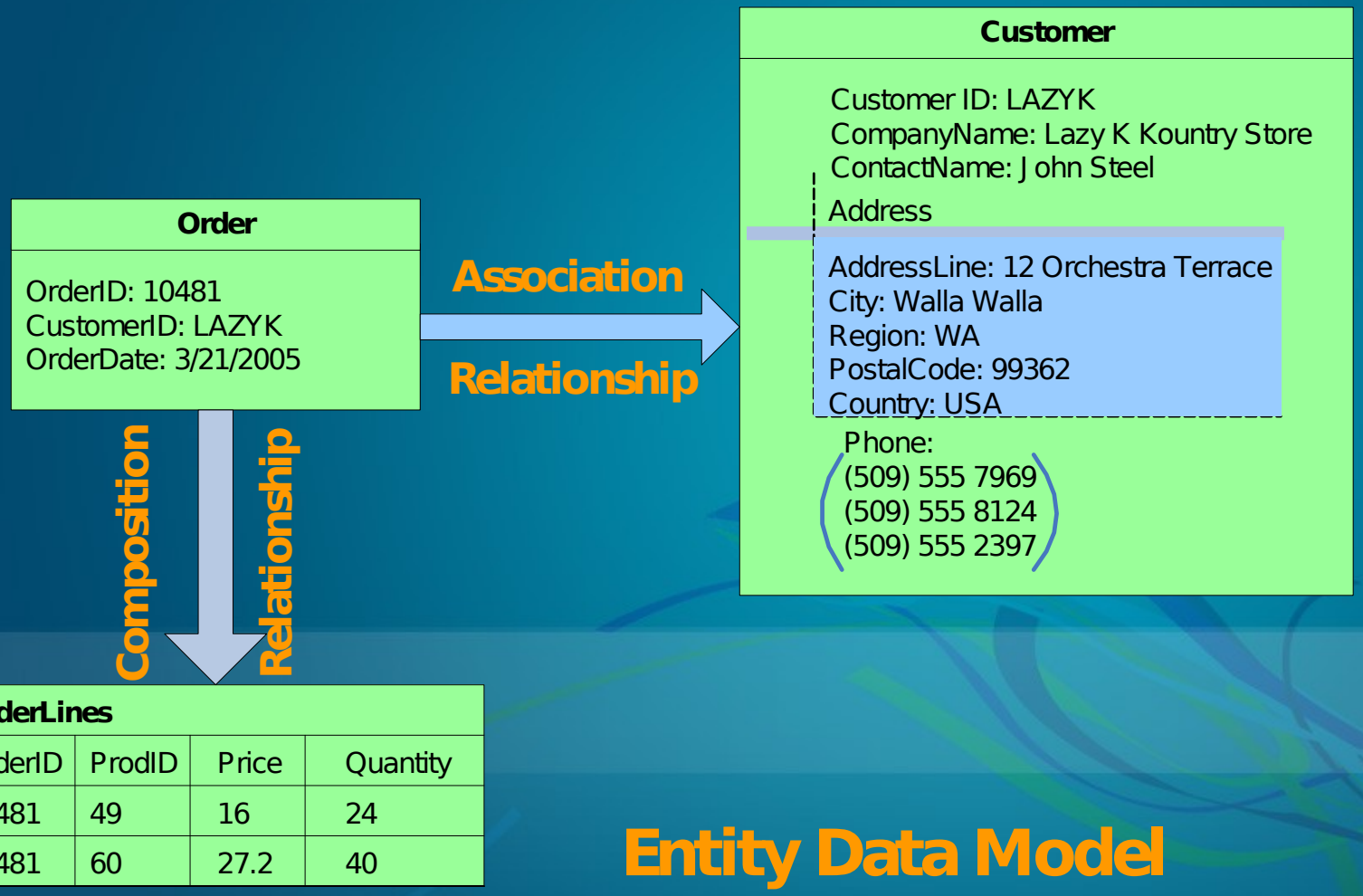
```
using System.Data.SqlClient;
using System.Data.CommandTree;

// Create a SqlConnection
SqlConnection sc = new
  SqlConnection(connStr);
sc.Open();

// Query for customers with orders >$100
QueryExpression qe =
  new QueryExpression(
  "Select Customer from Customers where
  Customer.Order.Amount > 100")
SqlCommand customers sc.CreateCommand();
customers.SetCommand(qe);

// Iterate through results
foreach(DataRow row in customers)
  Console.WriteLine(row.GetString(0);

// Add a new customer
SqlCommand cmd = sc.CreateCommand();
cmd.SetCommand(myUpdateTree);
cmd.ExecuteNonQuery();

// Close Connection
sc.Close();
```



What We Hear…

- “I want to leverage conceptual schemas in my database apps…”
  - Flexibility to insulate my code from db schema that doesn’t fit my application
  - Give me tools to accommodate version differences in the storage layout
  - Let my analysis applications use a different schema layout than my TP apps

Data Model In Brief



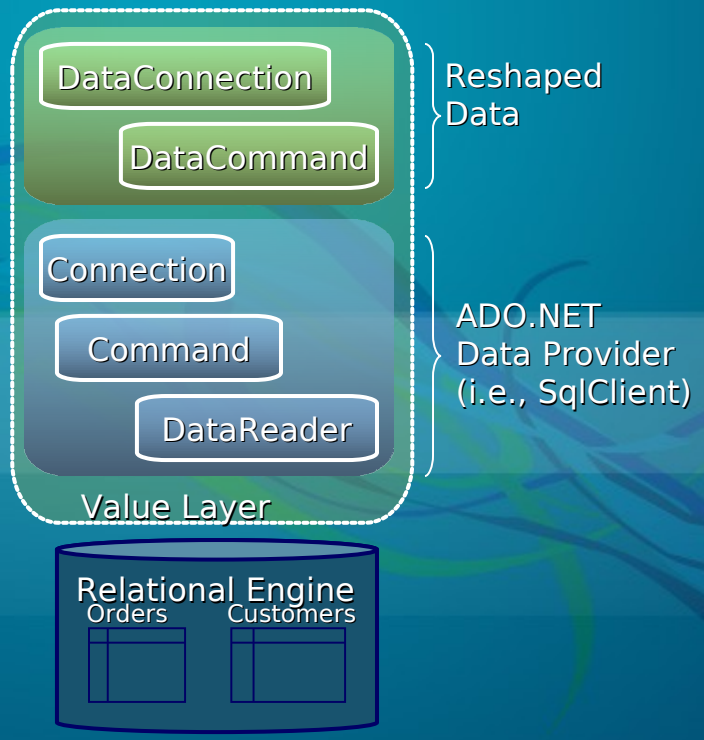
Entity Data Model

Why Have A Data Model?

- Everyone already has a de facto conceptual mode
- It provides a component model for services “close to the data”
- Provides a formal foundation for the underlying query semantics
  - Extend relational with a few key concepts

Flexible Storage Mapping

```
using System.Data.Common;

// Create a Mapped Connection
IStorageMapping smap =
  new MappingFile(myStoreMapping);
DataConnection dc =
  new Dataconnection(
          new SqlConnection(connStr),
          smap);

// Query for customers named "Smith"
DataCommand customers =
  dc.CreateCommand();
dc.CommandText = myOSQLstring;

// Iterate through results
foreach (DataRow row in customers)
  Console.WriteLine(row.GetString(0));

// Add a new customer
DataCommand cmd = dc.CreateCommand();
cmd.CommandText =
"INSERT INTO Customers (ID, LastName)
  VALUES ('00', 'Ellis')";
cmd.ExecuteNonQuery();
```

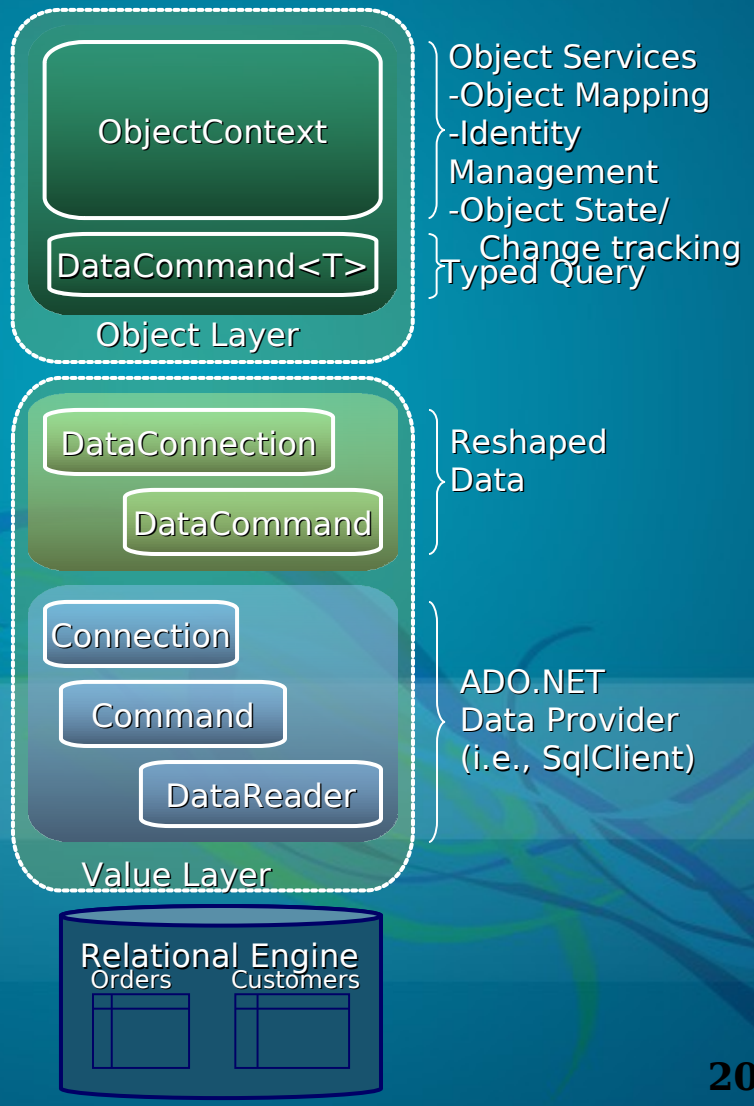


What We Hear…

- “I want a better transition from persistent tables to run-time objects for my apps…”
  - Let my programmers stay in their object model
  - Free me from maintaining my own ad-hoc O/R mapping infrastructure
  - Type check data shapes at compile-time

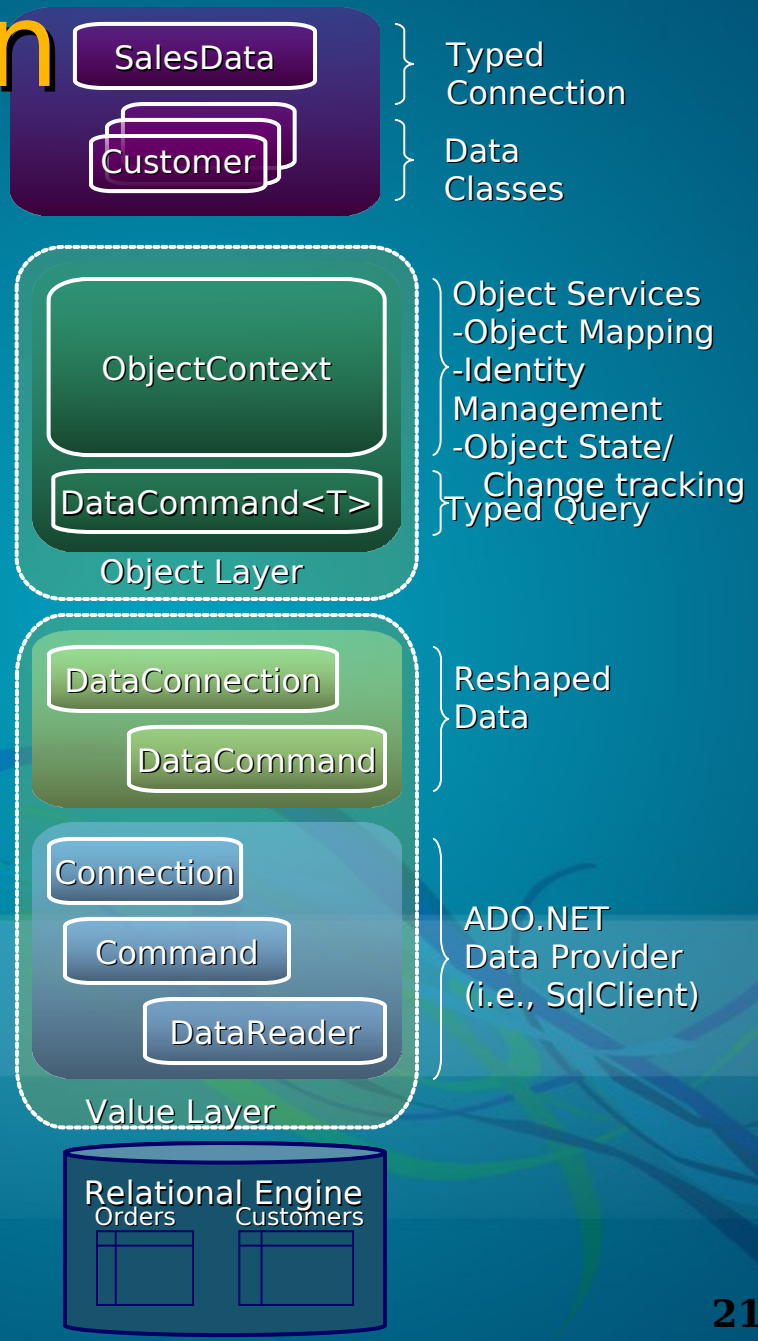
Object Mapping

```
using System.Data.Objects;

// Create a ObjectContext
// with explicit object mappings
IObjectMapping amap =
  new ObjectMappingFile(objectMapFile);
ObjectContext oc =
  new ObjectContext(myDataConfig,smap);

// Query for customers named "Smith"
DataCommand<Customers> customers =
  oc.CreateCommand<Customers>( “Customers
  ”).Where(“LastName=‘Smith’”);

// Iterate through results
foreach (Customer customer in customers)
  Console.WriteLine(Customer.LastName);

// Add a new customer
Customer c = new
  Customer(id,fname,lname);
oc.Add(“Customers”,c)
oc.SaveChanges();
```



Typed Connection

```
using System.Storage;
using System.Data.Objects;
using Druker.Sales;

// Create a typed SalesData Connection
SalesData sd = new
  SalesData(salesConfig);

// Query for customers named "Smith"
DataCommand<Customers> customers =
  sd.Customers.Where("LastName='Smith'");

// Iterate through results
foreach (Customer customer in customers)
  Console.WriteLine(customer.FirstName);

// Add a new customer
Customer customer =
  new Customer(id, fname, lname);
sd.Customers.Add(customer);
sd.SaveChanges();
```



What We Hear…

- “I want a language integrated query programming model for database apps…”
  - Remove impedance mismatch between my programming language and query language
  - Allow me to get compile-time analysis of my queries – help me reduce run-time errors
  - Leverage the richness of modern development environments
    - Intellisense on everything

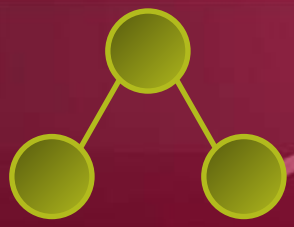
The LINQ Project

FUTURE TECHNOLOGIES

C# | VB | Others...

.NET Language Integrated Query

Standard Query Operators | DLinq (ADO.NET) | XLinq (System.Xml)



Objects

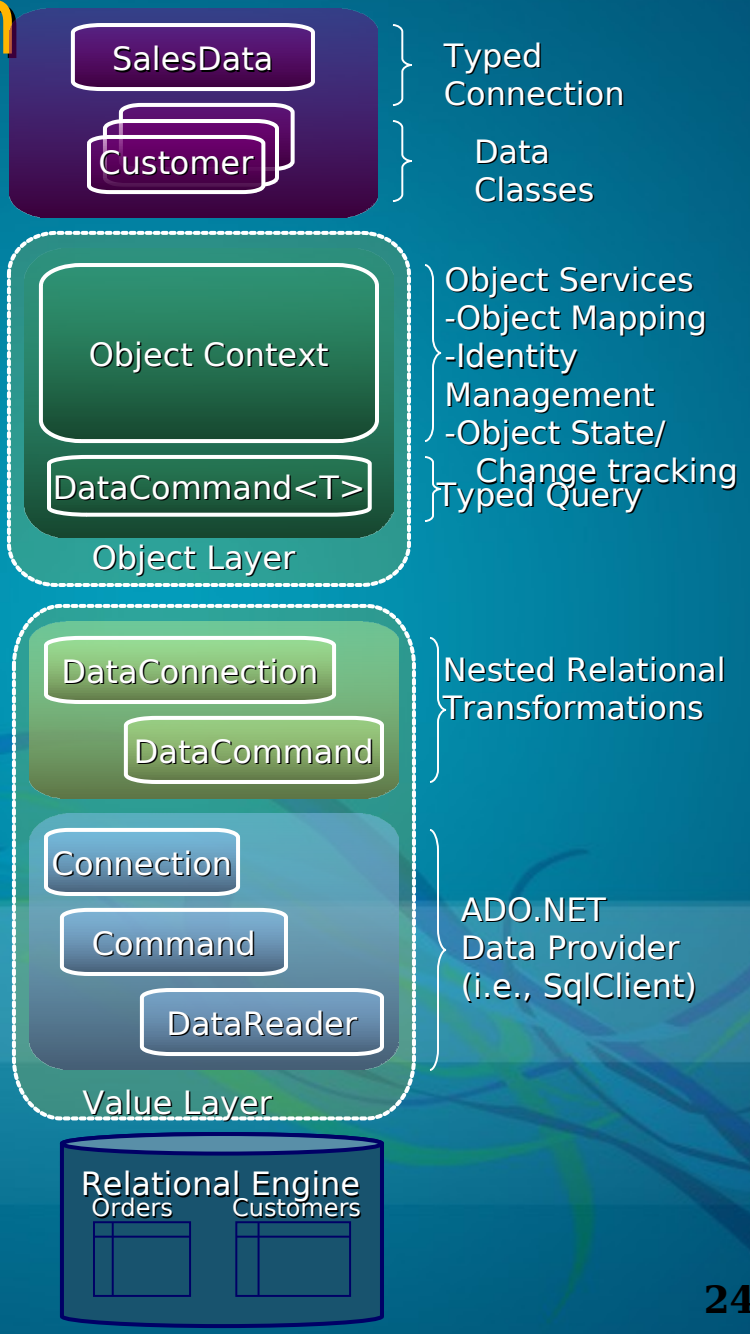
SQL

WinFS

XML

Language Integration

```
using System.Storage;
using System.Data.Objects;
using Druker.Sales;

// Create a typed SalesData Connection
SalesData sd = new SalesData(salesConfig);

// Iterate through customers named 'Smith'
foreach (Customer customer in sd.Customers
  where LastName='Smith')
  Console.WriteLine(customer.FirstName);

// Add a new customer
Customer customer =
  new Customer(id, fname, lname);
sd.Customers.Add(customer);
sd.SaveChanges();
```



What We Hear…

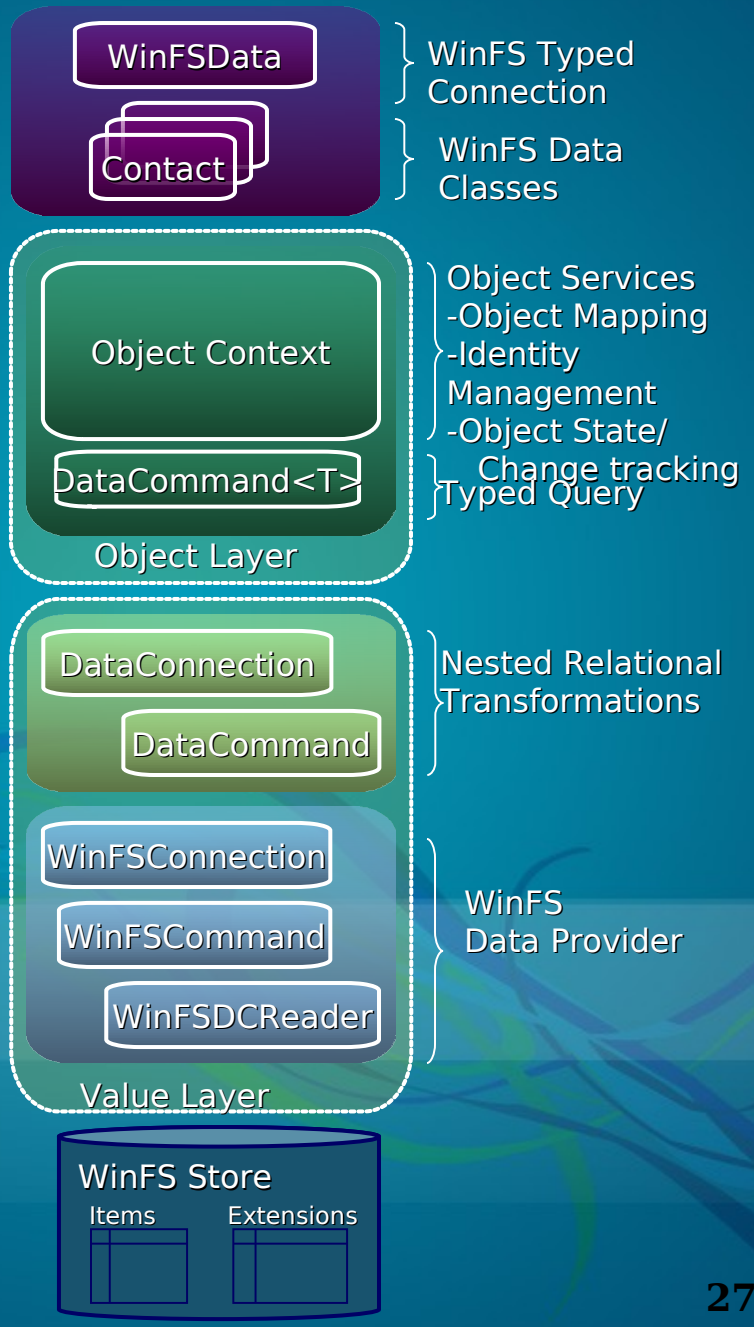
- “I want to build my own frameworks for extensibility on my database app…”
  - Let me provide extended services outside of CRUD/Query
  - Take full advantage of Microsoft tools and components
  - Give me an environment that helps programmers “get it right” in my application

Integrated Storage: WinFS

- The WinFS programming model is a library of services and schemas for client data platform
- Data model
  - Items, Links and Extensions are specializations of the entity data model nouns: Entity, Relationships and EntityExtensions
- Services
  - Synchronization, storage views for visualization, unifed backup and restore, win32 compatibility, stores and shares, full-text search, Lists and Autolists
- Everyday information schemas
  - Contact, Calendar, Message, Document, Image, Music, Task

WinFS Pattern

```
using System.Storage;
using System.Data.Objects;

// Create a typed WinFS Connection
WinFSData wd = new WinFSData("\\localhost\
  defaultstore");

// Query for contacts named "Smith"
DataCommand<contacts> contacts =
  wd.Items.FilterByType<Contacts>();

// Iterate through results
foreach (Contact contact in contacts)
  Console.WriteLine(Contact.LastName);

// Add a new contact
Contact c =
  new Contact(container, id, fname, lname);
wd.Items.Add(c);
wd.SaveChanges();
```



DEMO

WinFS Programming With LINQ

PDC05 DEVELOPER POWERED

Samuel Druker
Development Manager
SQL Server

What We Hear
“I want…”

- …better programming model for db apps
- …leverage conceptual schemas in my apps
- …integrated support for queries in my apps
- …better transition from tables to objects
- …language integrated query
- …my own frameworks for extensibility

Community Resources

- At PDC
  - For more information, go see
    - DAT323: Using the .NET Language Integrated Query Framework with Relational Data (Fri 8:30)
    - DAT408 – ADO.NET 2.0: Advanced Data Access Patterns (Thu 10)
    - Labs: DATHOL07 Getting Started with WinFS
- After PDC
  - If you missed these, check out the DVD
    - DAT209: “WinFS” Future Directions: An Overview
    - DAT310: "WinFS" Future Directions: Building Data Centric Applications Using Windows Presentation Foundation ("Avalon") and Windows Forms
  - Team blog: http://blogs.msdn.com/winfs
  - Newsgroups microsoft.public.windows.developer.winfx.winfs
  - Channel 9: http://channel9.msdn.com/tags/WinFS

Microsoft®

Your potential. Our passion.™